

# Python: module eof.analyzer

***eof.analyzer***

[index](#)  
[/pcmdi/halliday1/cdat-4.0/lib/python2.4/site-packages/rs\\_dir/eof/analyzer.py](/pcmdi/halliday1/cdat-4.0/lib/python2.4/site-packages/rs_dir/eof/analyzer.py)

## Modules

<a href="#">LinearAlgebra</a>	<a href="#">copy_reg</a>	<a href="#">pickle</a>	<a href="#">types</a>
<a href="#">MLab</a>	<a href="#">math</a>	<a href="#">eof.rs</a>	
<a href="#">copy</a>	<a href="#">multiarray</a>	<a href="#">string</a>	

## Classes

[Analyzer](#)

class ***Analyzer***

Methods defined here:

***analyze***(self, z, nr=0, space=None)

## Functions

***arange***(...)

`arange(start, stop=None, step=1, typecode=None)`

Just like `range()` except it returns an array whose type can be specified by the keyword argument `typecode`.

***array***(...)

`array(sequence, typecode=None, copy=1, savespace=0)` will return a

***arrayrange*** = `arange(...)`

`arange(start, stop=None, step=1, typecode=None)`

Just like `range()` except it returns an array whose type can be specified by the keyword argument `typecode`.

***choose***(...)

`choose(a, (b1,b2,...))`

***cross\_correlate***(...)

`cross\_correlate(a,v, mode=0)`

***eigsort***(d, v, nr)

Sort the eigenvalues and vectors, return first nr of them

***fromstring***(...)

fromstring(string, typecode='l', count=-1) returns a new 1d array

***reshape***(...)

reshape(a, (d1, d2, ..., dn)). Change the shape of a to be an n-d

***searchsorted*** = binarysearch(...)

binarysearch(a, v)

***take***(...)

take(a, indices, axis=0). Selects the elements in indices from a

***zeros***(...)

zeros((d1, ..., dn), typecode='l', savespace=0) will return a new array

## Data

***Character*** = 'c'

***Complex*** = 'D'

***Complex0*** = 'F'

***Complex16*** = 'F'

***Complex32*** = 'F'

***Complex64*** = 'D'

***Complex8*** = 'F'

***Error*** = 'EOF Error.'

***Float*** = 'd'

***Float0*** = 'f'

***Float16*** = 'f'

***Float32*** = 'f'

***Float64*** = 'd'

***Float8*** = 'f'

***Int*** = 'l'

***Int0*** = '1'

***Int16*** = 's'

***Int32*** = 'i'

***Int8*** = '1'

***LittleEndian*** = True

***NewAxis*** = None

***PyObject*** = 'O'

***UInt*** = 'u'

***UInt16*** = 'w'

***UInt32*** = 'u'

***UInt8*** = 'b'

***UnsignedInt16*** = 'w'

***UnsignedInt32*** = 'u'

***UnsignedInt8*** = 'b'

***UnsignedInteger*** = 'u'

***absolute*** = <ufunc 'absolute'>

***add*** = <ufunc 'add'>  
***arccos*** = <ufunc 'arccos'>  
***arccosh*** = <ufunc 'arccosh'>  
***arcsin*** = <ufunc 'arcsin'>  
***arcsinh*** = <ufunc 'arcsinh'>  
***arctan*** = <ufunc 'arctan'>  
***arctan2*** = <ufunc 'arctan2'>  
***arctanh*** = <ufunc 'arctanh'>  
***bitwise\_and*** = <ufunc 'bitwise\_and'>  
***bitwise\_or*** = <ufunc 'bitwise\_or'>  
***bitwise\_xor*** = <ufunc 'bitwise\_xor'>  
***ceil*** = <ufunc 'ceil'>  
***conjugate*** = <ufunc 'conjugate'>  
***cos*** = <ufunc 'cos'>  
***cosh*** = <ufunc 'cosh'>  
***divide*** = <ufunc 'divide'>  
***divide\_safe*** = <ufunc 'divide\_safe'>  
***e*** = 2.7182818284590451  
***equal*** = <ufunc 'equal'>  
***exp*** = <ufunc 'exp'>  
***fabs*** = <ufunc 'fabs'>  
***floor*** = <ufunc 'floor'>  
***floor\_divide*** = <ufunc 'floor\_divide'>  
***fmod*** = <ufunc 'fmod'>  
***greater*** = <ufunc 'greater'>  
***greater\_equal*** = <ufunc 'greater\_equal'>  
***hypot*** = <ufunc 'hypot'>  
***invert*** = <ufunc 'invert'>  
***left\_shift*** = <ufunc 'left\_shift'>  
***less*** = <ufunc 'less'>  
***less\_equal*** = <ufunc 'less\_equal'>  
***log*** = <ufunc 'log'>  
***log10*** = <ufunc 'log10'>  
***logical\_and*** = <ufunc 'logical\_and'>  
***logical\_not*** = <ufunc 'logical\_not'>  
***logical\_or*** = <ufunc 'logical\_or'>  
***logical\_xor*** = <ufunc 'logical\_xor'>  
***maximum*** = <ufunc 'maximum'>  
***minimum*** = <ufunc 'minimum'>  
***multiply*** = <ufunc 'multiply'>  
***negative*** = <ufunc 'negative'>  
***not\_equal*** = <ufunc 'not\_equal'>  
***pi*** = 3.1415926535897931  
***power*** = <ufunc 'power'>  
***remainder*** = <ufunc 'remainder'>  
***right\_shift*** = <ufunc 'right\_shift'>  
***sin*** = <ufunc 'sin'>  
***sinh*** = <ufunc 'sinh'>  
***sqrt*** = <ufunc 'sqrt'>  
***subtract*** = <ufunc 'subtract'>  
***tan*** = <ufunc 'tan'>  
***tanh*** = <ufunc 'tanh'>



```
true_divide = <ufunc 'true_divide'>
```

```
typecodes = {'Character': 'c', 'Complex': 'FD', 'Float': 'fd', 'Integer': 'l sil', 'UnsignedInteger': 'bwu'}
```